

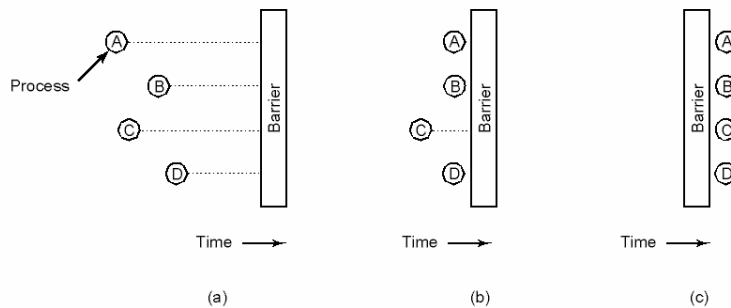
More IPC

B.Ramamurthy

9/28/2006

1

Barriers



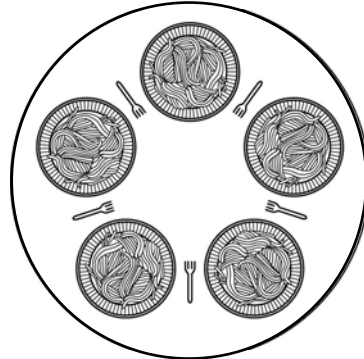
- Use of a barrier
 - processes approaching a barrier
 - all processes but one blocked at barrier
 - last process arrives, all are let through

9/28/2006

2

Dining Philosophers (1)

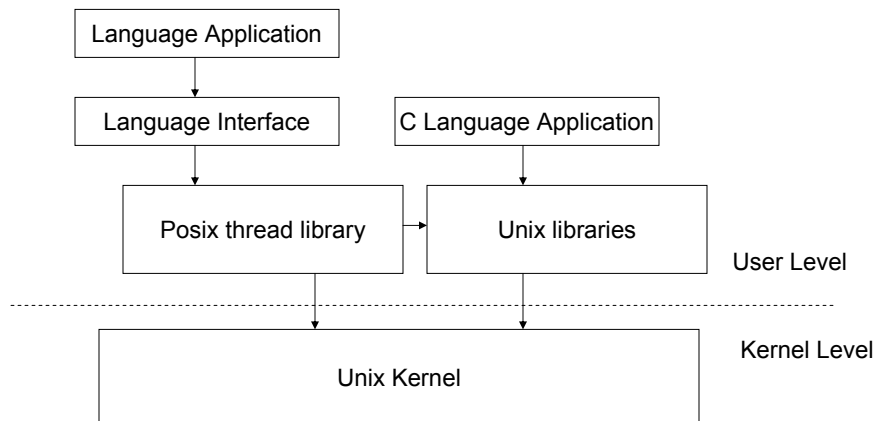
- Philosophers eat/think
- Eating needs 2 forks
- Pick one fork at a time
- How to prevent deadlock



9/28/2006

3

Posix Library Implementation Based F. Mueller's Paper



To answer Hanumanth's question.

9/28/2006

4

Dining Philosophers Example

```
monitor dp
{
    enum {thinking, hungry, eating} state[5];
    condition self[5];
    void pickup(int i)           // following
slides
    void putdown(int i)       // following slides
    void test(int i)         // following slides
    void init() {
        for (int i = 0; i < 5; i++)
            state[i] = thinking;}
}
```

9/28/2006

5

Dining Philosophers

```
void pickup(int i) {
    state[i] = hungry;
    test[i];
    if (state[i] != eating)
        self[i].wait();
}

void putdown(int i) {
    state[i] = thinking;
    // test left and right neighbors
    test((i+4) % 5);
    test((i+1) % 5);
}
```

9/28/2006

6

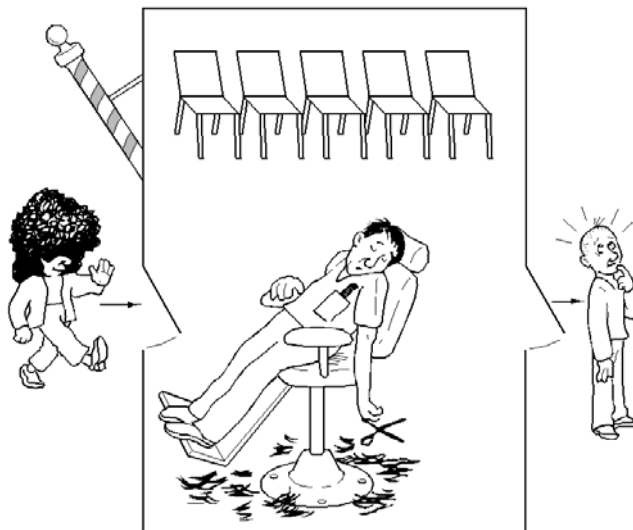
Dining Philosophers

```
void test(int i) {  
    if ( (state[(i + 4) % 5] != eating) &&  
        (state[i] == hungry) &&  
        (state[(i + 1) % 5] != eating)) {  
        state[i] = eating;  
        self[i].signal();  
    }  
}
```

9/28/2006

7

The Sleeping Barber Problem (1)



9/28/2006

8

Sleeping Barber Problem

The barber shop has one barber, one barber chair, and n waiting chairs.

If there are no customers, barber falls asleep. When customer arrives he/she has to wake up sleeping barber.

If additional customer arrive when barber is cutting they wait (if any empty chair) or leave (if all chairs are taken).